PORTFOLIO WORK SAMPLE - DATA QUALITY

Data Quality Profiler (SQL)

SQL - profiling queries with sample output (PostgreSQL)

Sanitized work sample authored by Bernard Millet. All client-specific data, names and identifiers have been removed; it is representative of approach and craft, not a client deliverable.

```
/* ========================================================================
   DATA QUALITY PROFILER: a simple, worked example (PostgreSQL)
   Author: Bernard Millet. DAMA-DMBOK: Data Quality Management
   Portfolio work sample. Sanitized and representative; no proprietary data.
   ------------------------------------------------------------------------
   WHAT THIS SCRIPT DOES
     1. PROFILING  -> describes the data (rows, missing values, distinct
                      values, ranges). It answers "what is in this table?"
     2. DQ CHECKS  -> counts rule violations. It answers "is the data good?"

   This file is self-contained: it builds a tiny sample table with DELIBERATE
   problems, then runs both queries on it. Run the whole file on PostgreSQL
   and you will get the results shown in the comments.
   ======================================================================== */

-- ------------------------------------------------------------------------
-- 0. SAMPLE DATASET  (invented; the problems are flagged in the comments)
-- ------------------------------------------------------------------------
DROP TABLE IF EXISTS customers;
CREATE TEMP TABLE customers (
    customer_id  text,
    name         text,
    email        text,
    country_code text,
    created_at   date,
    updated_at   date
);

INSERT INTO customers VALUES
  ('C001','Alice Martin','alice.martin@example.com','FR','2025-01-10','2025-01-12'),
  ('C002','Bob Smith',   'bob.smith@example.com',   'DE','2025-02-03','2025-02-05'),
  ('C003','Carla Rossi', 'carla@example.com',       'IT','2025-02-20','2025-02-19'), -- country IT not allowed; updated < created
  ('C004','David Lee',   'not-an-email',            'GB','2025-03-01','2025-03-02'), -- email has no @
  ('C004','David Lee',   'david.lee@example.com',   'GB','2025-03-01','2025-03-05'), -- duplicate id C004
  ('C006','Emma Dubois',  NULL,                     'FR','2025-03-10','2025-03-11'), -- email missing
  ('C007','Frank Meyer', 'frank.meyer@example.com', 'CH','2025-03-15','2025-03-16'),
  ('C008','Grace Kim',   'grace.kim@example.com',   'US','2025-04-01','2025-04-02'),
  (NULL,  'Henry Ford',  'henry.ford@example.com',  'BE','2025-04-05','2025-04-06'), -- customer_id missing
  ('C010','Ivy Chen',    'ivy.chen@example',        'FR','2025-04-08','2025-04-09'); -- email has no domain ending


-- ------------------------------------------------------------------------
-- 1. PROFILING  -  describe the table (no rules, just facts)
-- ------------------------------------------------------------------------
-- COUNT(col) ignores NULLs, so "filled" = COUNT(col) and "missing" = total - that.
SELECT
    COUNT(*)                            AS row_count,
    COUNT(customer_id)                  AS id_filled,
    COUNT(*) - COUNT(customer_id)       AS id_missing,
    COUNT(DISTINCT customer_id)         AS id_distinct,
    COUNT(email)                        AS email_filled,
    COUNT(*) - COUNT(email)             AS email_missing,
    COUNT(DISTINCT country_code)        AS country_distinct,
    MIN(created_at)                     AS earliest_created,
    MAX(updated_at)                     AS latest_updated
FROM customers;

/* EXPECTED RESULT
 row_count | id_filled | id_missing | id_distinct | email_filled | email_missing | country_distinct | earliest_created | latest_updated
-----------+-----------+------------+-------------+--------------+---------------+------------------+------------------+----------------
        10 |         9 |          1 |           8 |            9 |             1 |                7 | 2025-01-10       | 2025-04-09
   Reading it: 10 rows; 1 customer_id missing; 8 distinct ids out of 9 filled (so 1 duplicate);
   1 email missing; 7 distinct countries.
*/


-- ------------------------------------------------------------------------
-- 2. DATA QUALITY CHECKS  -  count how many rows break each rule
-- ------------------------------------------------------------------------
-- COUNT(*) FILTER (WHERE ...) counts only the rows that match the condition.
SELECT 'Completeness: customer_id present' AS rule,
       COUNT(*) FILTER (WHERE customer_id IS NULL) AS failing_rows FROM customers
UNION ALL SELECT 'Completeness: email present',
       COUNT(*) FILTER (WHERE email IS NULL) FROM customers
UNION ALL SELECT 'Uniqueness: customer_id (surplus duplicates)',
       COUNT(customer_id) - COUNT(DISTINCT customer_id) FROM customers
UNION ALL SELECT 'Validity: email format',
       COUNT(*) FILTER (WHERE email IS NOT NULL
                        AND email !~ '^[^@\s]+@[^@\s]+\.[^@\s]+$') FROM customers
```

```
UNION ALL SELECT 'Validity: country on allowed list',
       COUNT(*) FILTER (WHERE country_code NOT IN ('FR','DE','BE','LU','CH','GB','US')) FROM customers
UNION ALL SELECT 'Consistency: created_at <= updated_at',
       COUNT(*) FILTER (WHERE created_at > updated_at) FROM customers;

/* EXPECTED RESULT
 rule                                          | failing_rows
-----------------------------------------------+--------------
 Completeness: customer_id present             |            1   (the NULL id row)
 Completeness: email present                   |            1   (Emma Dubois)
 Uniqueness: customer_id (surplus duplicates)  |            1   (second C004)
 Validity: email format                        |            2   (not-an-email, ivy.chen@example)
 Validity: country on allowed list             |            1   (Carla Rossi, IT)
 Consistency: created_at <= updated_at         |            1   (Carla Rossi)
*/
```